Statistical Anomaly Detection for Train Fleets

Anders Holst and **Markus Bohlin** and **Jan Ekman**
Swedish Institute of Computer Science
Box 1263, SE-164 29 Kista, Sweden

Ola Sellin
Bombardier Transportation
Östra Ringvägen 2, SE-721 23 Västerås, Sweden

Björn Lindström[1] and **Stefan Larsen**[2]
[1] Addiva Consulting AB [2] Addiva Eduro AB
Kopparbergsvägen 8, SE-722 13 Västerås, Sweden

Abstract

We have developed a method for statistical anomaly detection which has been deployed in a tool for condition monitoring of train fleets. The tool is currently used by several railway operators over the world to inspect and visualize the occurrence of "event messages" generated on the trains. The anomaly detection component helps the operators to quickly find significant deviations from normal behavior and to detect early indications for possible problems. The savings in maintenance costs comes mainly from avoiding costly breakdowns, and have been estimated to several million Euros per year for the tool. In the long run, it is expected that maintenance costs can be reduced with between 5 and 10 % by using the tool.

Introduction

Anomaly detection is a growing area with more and more practical applications every day. It has been used for fraud detection and intrusion detection for a long time, but in later years the usage has exploded to all kind of domains, like surveillance, industrial system monitoring, epidemiology, etc. For an overview of different anomaly detection methods and applications, see e.g. [Chandola, Banerjee, and Kumar, 2009].

The approach taken in *Statistical anomaly detection* is to use data from (predominantly normal) previous situations to build a statistical model of what is normal. New situations are compared against that model, and are considered anomalous if they are too improbable to occur in that model.

Addtrack is a tool for condition monitoring of trains developed in collaboration between Bombardier Transportation AB and Addiva AB. Addtrack is currently used by Bombardier and several railway operators to inspect and visualize the occurrence of "event messages" generated on train fleets. The data sets analyzed often consists of several thousand data points, and a common complaint from analysts was that it was, as a consequence, difficult to observe patterns and anomalies in the fleet of trains. Of particular interest was the ability to more directly point out the most anomalous observations, so that further investigative actions could be taken.

SICS has for several years developed methods for statistical anomaly detection, based on a framework called "Bayesian Principal Anomaly" [Holst and Ekman, 2011]. The framework has already been successfully used and evaluated in several real application domains, such as alarm filtering in telecommunication networks, intrusion detection [Dey, 2009], alarm call monitoring for crisis management, and maritime domain awareness [Bjurling et al., 2010]. In this paper we describe a novel application domain for the anomaly detection method; condition monitoring of trains [Holst, Ekman, and Larsen, 2006]. The developed methods have been deployed as a component of the Addtrack system, and are in use helping analysts to quickly find significant deviations from normal behavior in several train fleets, and to detect early indications for possible problems.

Other Bayesian methods have previously been applied to a wide variety of problems including intrusion detection [García-Teodoro et al., 2009; Cemerlic, Yang, and Kizza, 2008; Chebrolu, Abraham, and Thomas, 2005], dynamic system diagnosis [Lerner et al., 2000], spam filtering [Su and Xu, 2009], environmental anomaly detection [Hill, Minsker, and Amir, 2009] and energy expenditure estimation [Shahabdeen, Baxi, and Nachman, 2010]. However, applications to train diagnosis have, as far as we know, not been published before.

The rest of this paper is organized as follows. First, the Addtrack tool is described together with the original requirements for an anomaly detection module and an outline of how the application is used. Next, the principal anomaly is presented as the basis for the module, followed by a description of the Bayesian learning used to estimate the normality model and the particular application to event data anomaly detection. The anomaly detection module is then described, and the development and deployment process follows. The paper ends with a brief evaluation and a discussion of the results and future development.

The Addtrack Tool

Addtrack is a tool developed originally by Bombardier Transportation for general analysis, monitoring and visualization of train condition and event data, which is periodically or continuously uploaded from the train units. It is "intelligent" in the sense that analysis modules, such as the one described in this paper, can be used to preprocess and vi-

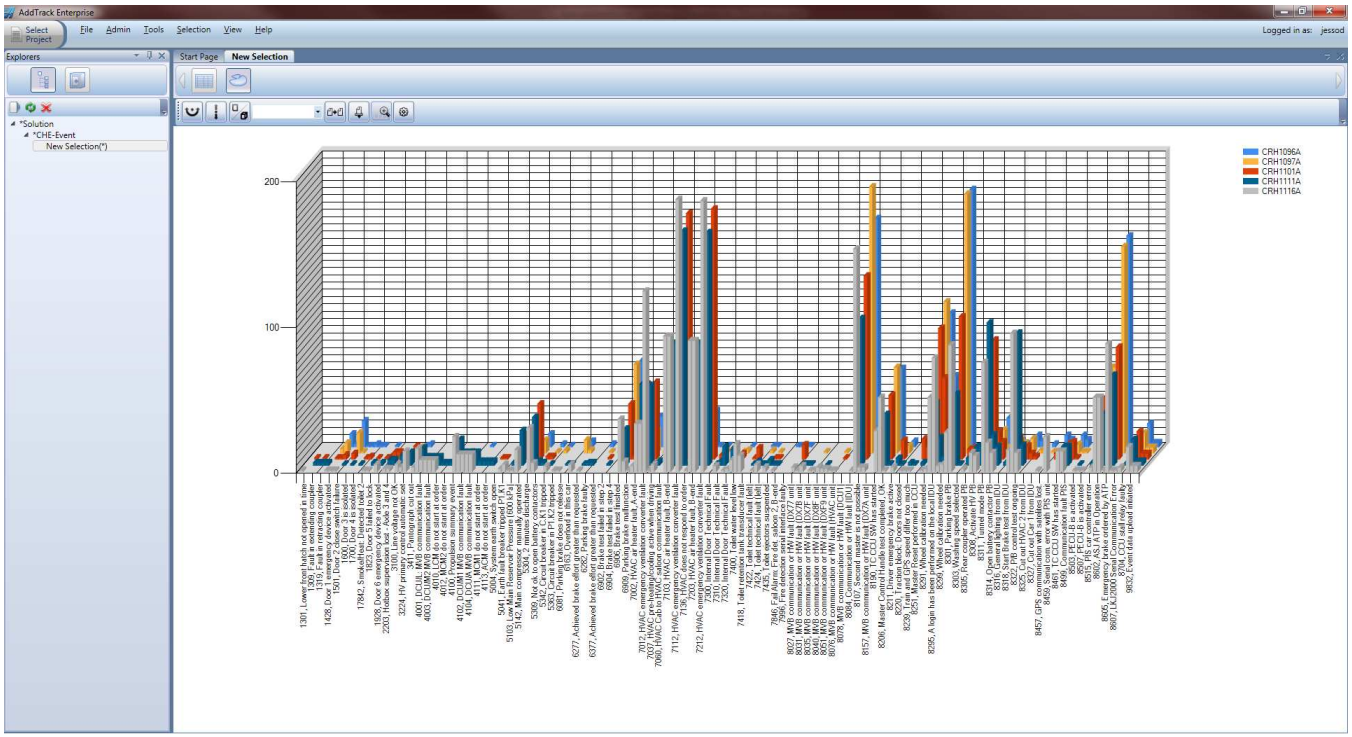

Figure 1: Addtrack visualization of selected event occurrences for five units in the Chinese fleet.

sualize data sets. Today, Addtrack is developed by the independent company Addiva Consulting AB. Addtrack including the anomaly detection module described in this paper is currently deployed in Sweden, India, China and Germany, and has approximately 270–300 active users in total. Out of these, most use the anomaly detection functionality. The main purpose of Addtrack is both to let the maintenance personel get an overview of the condition of the fleet to catch problems early, and also to find out the circumstances and causes when a failure has occurred.

The primary user of Addtrack is Bombardier Transportation, one of the world's largest rail-equipment manufacturing and servicing companies with more than 100 000 installed rail cars and locomotives worldwide. Here, the main use of Addtrack is to detect faulty units during the development of new railway vehicles and the subsequent commissioning phase. Following this, Addtrack is used during the guarantee period to perform a simple form of root-cause analysis. Finally, the tool is used during the maintenance phase for acute fault localization. To a lesser extent, it is also used for predictive maintenance, but this mode of use is growing since it allows the operators to catch faults early, thereby avoiding much more severe stopping failures. Today, customers using Addtrack prevent on average one or two stopping faults annualy for each train unit.

A key functionality of Addtrack is to visualize the number of events of different types that have occurred on a set of trains during some period. Figure 1 contains such a view, where the diagram shows the number of events per event code (on the x-axis) and train (on the z-axis) that have occurred during the last month. Only event codes that have occurred at all during the month are shown. In the menu to the left it is possible to select which set of trains to show. It is also possible to click a bar in the diagram to see a time series of which days those events occurred (not shown here).

As can be seen, there are rather many event codes with a high number of events for most trains, indicating that this is the normal state. There is also a high variability between trains in the number of events of some event codes, but it is not obvious from the diagram when a bar is (statistically) significantly higher than the others. Before the integration of the anomaly detection module, analysts were analyzing the raw fault rates exclusively by choosing a period and a subset of trains and codes for events or condition data to visualize. It is however a time consuming and tedious work for an analyst to go through all high bars to see if they indicate something out of the ordinary. There may even be quite low bars that nevertheless are significantly higher than for other trains, but may go unnoticed among all the normally high bars. Indeed, sometimes it may even be remarkable with a too low number of events, which is almost impossible to spot in the diagram.

As an example of the size of the data set, for the Swedish fleet of Regina trains operated by SJ, which consists of 57 different train sets, there is over 1 000 different event and condition types. The data is typically collected weekly, and for each such interval the analyst therefore have a set of over 57 000 data points to watch in total. On the more modern train sets, over 12 000 data points are collected per train set. The main complaint of analysts was therefore that it was

very difficult to observe patterns in the huge amounts of data that the analyst has to handle. A particular problem was the analysis of rare events, for which obvious patterns was hard to detect even for experienced analysts.

In summary, it would be advantageous with a tool to indicate which bars are significantly different from expected, and should thus be focused on. To be useful in practice, it was also judged important that an anomaly detector module for Addtrack fulfill several conditions:

- It must be able to handle a large number of input features.
- It must be able to handle several different normal situations.
- It must allow for training data to include realistic amounts of anomalous cases.
- It must be fast when dealing with large amounts of data.
- It must be robust in the light of very small amounts of training data.
- It must have a sufficiently small false alarm rate, while still being maximally sensitive to real anomalies.

The following two sections describe an anomaly detection module designed from these requirements.

Principal Anomaly Detection

The general idea in statistical anomaly detection is to build a statistical model over normal cases, and then compare new samples with this model when they arrive. Samples that would have a too small probability of being generated by the statistical model are considered anomalous, i.e. they are very unlikely to belong to the set of normal cases. This is so far similar to classical hypothesis testing in that we are trying to reject the hypothesis that a new sample is generated by the normality model.

However, there is also a Bayesian statistics part, which comes in when estimating the parameters of the normality model. This estimation must be robust also when there are very few data samples. A classical maximum likelihood point estimate of the parameters can make the estimated distribution sensitive to random fluctuations in the data, which tends to result in too many false alarms. Using a Bayesian approach, we can remedy this by taking appropriate consideration to the uncertainties in the actually observed data, and thereby avoiding some of the false alarms.

Let us initially assume that the normal situation samples are generated by a known probability density $P(x \mid \theta)$ for a set of parameters θ. To define what is meant by an "anomalous" observation, we note that intuitively, the smaller the probability of generating a new observation z from the distribution, the more anomalous is it. However, just looking at the probability $P(z \mid \theta)$ of generating the observation z from the distribution won't do, since the magnitude of $P(z \mid \theta)$ depends on the variance of the distribution (for continuous distributions). Therefore, it is not possible to specify a fixed probability threshold that is the same for all distributions, and below which a sample should be considered anomalous. Instead, we should look at the same entity as in hypothesis testing, i.e. the probability of generating an observation *at least* as unusual as z. This also gives a natural way of controlling the rate of false alarms, which may otherwise be high in many anomaly detection applications.

Thus, let us then define the *principal anomaly* of a new observation z as the probability of generating a more common sample than z from the distribution:

$$A(z \mid \theta) = \int_{x \in \Omega} P(x \mid \theta) \quad (1)$$
$$\text{where } \Omega = \{x : P(x \mid \theta) > P(z \mid \theta)\}$$

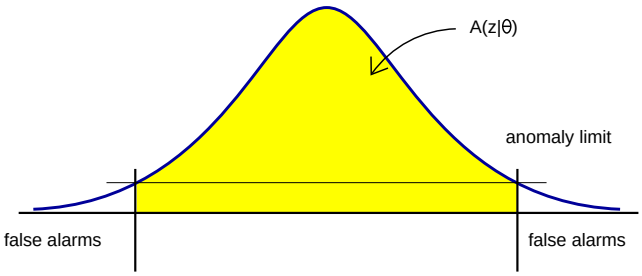

Figure 2: Illustration of the principal anomaly $A(z \mid \theta)$.

This measure is illustrated in Figure 2, and has a number of desirable properties. First, $A(z \mid \theta)$ increases when the sample z gets more unusual, which is intuitive. Second, it is directly comparable to the principal anomaly of other distributions or sets of parameters. Third, it is directly connected to the rate of false alarms. If we set a threshold on the principal anomaly of $1 - \epsilon$ over which an observation is judged anomalous, the probability that a normal sample is wrongly detected as anomalous is then simply ϵ.

In the context of statistical anomaly detection, we claim that all sound scores of anomaly should be related to this principal anomaly, since it defines how unusual a sample is. Conversely, any score that is a monotonic function of the principal anomaly will rank the samples identically with respect to anomaly.

However, although suitable in the formal sense, the principal anomaly itself is not so convenient to work with and inspect manually, since its anomaly values are most of the time very close to 1. Its complement may therefore be more useful:

$$\bar{A}(z \mid \theta) = 1 - A(z \mid \theta) = \int_{x \in \bar{\Omega}} P(x \mid \theta) \quad (2)$$
$$\text{where } \bar{\Omega} = \{x : P(x \mid \theta) \leq P(z \mid \theta)\}$$

This is the probability of getting an equally or less probable sample than z from the distribution, i.e. the probability of the tails beyond z (and beyond other samples with the same probability as z), and therefore equal to ϵ. For anomalous samples it will be very close to 0, and therefore easier to compute with high precision than the principal anomaly itself. Also useful, and more intuitive to work with, is the negative logarithm of the complementary principal anomaly:

$$\Lambda(z \mid \theta) = -\log(\bar{A}(z \mid \theta)) \quad (3)$$

It ranges from 0 to ∞, and increases with higher anomaly, such that each constant step higher represents a factor lower probability. But as mentioned above, exactly which transform is used for presentation purposes is irrelevant, as long as it is strictly monotonically increasing in the principal anomaly A.

Bayesian Learning

Let us now consider the case when the parameters θ are unknown, and we instead have to estimate the result from a number of samples. To be useful, the anomaly detector should be able to provide an anomaly score already after a very small number of training samples. With maximum likelihood estimation of the parameters, the estimates will be far too sensitive to chance occurrences, and the resulting detector will tend to strongly underestimate the probability of new samples, resulting in too many false alarms. With a Bayesian approach, all parameter values that may have given rise to the observed samples are considered (appropriately weighted). The result is a detector that is much more robust to random fluctuations in the training data, and doesn't indicate an anomaly unless it is sufficiently certain. The side effect is that early on a Bayesian anomaly detector will accept more samples as normal. However, as more training data is collected, the parameter estimation will become more accurate which in turn will make the anomaly detector more precise.

More formally, the Bayesian approach is thus to find the posterior distribution over the parameters θ given the set of training samples X:

$$P(\theta \mid X) \propto P(X \mid \theta)P(\theta) = \prod_i P(x_i \mid \theta)P(\theta) \quad (4)$$

Here, $P(\theta)$ is the prior distribution over the parameters. In this paper we use a standard non-informative prior which has proven adequate in practice. We can now obtain the principal anomaly by integration over all possible parameter values:

$$A(z \mid X) = \int_\theta A(z \mid \theta)P(\theta \mid X) \quad (5)$$

We define this as the *Bayesian Principal Anomaly*.

Anomalies in Event Data

In the train condition monitoring application, the data used to check for anomalies are the rates of different event messages generated on the trains. As on most technical systems, there is a large amount of log messages generated when things happen, representing events ranging from harmless to serious. The serious events are straightforward to handle, since they typically require service more or less immediately. More interesting from an anomaly detection perspective are changes in the rates of less serious or seemingly harmless events. Such often subtle changes in the rates nevertheless indicates that something has changed on the train, which may merit action in the form of an extra inspection of the corresponding subsystem.

To model normal behaviour, we assume that when the train is in a normal state, each event type has a certain normal rate with which it occurs. Under this assumption, we can model the number of events in an interval of length T with a Poisson distribution:

$$P(x \mid \lambda T) = (\lambda T)^x e^{-\lambda T}/x! \quad (6)$$

where λ is the rate of events per time unit. For each event type x_i, we get the Bayesian Principal Anomaly from Equation (5) by inserting the above expression with $\theta = \lambda T$ in Equation (4). The resulting expression requires numerical evaluation of an indefinite sum, but not more complicated than that it can be computed rather fast on a normal computer.

It is now possible to test each train against the others, by counting the number of events of each type for each train during a certain time period of interest, and computing the Bayesian Principal Anomaly for the counts of each train, basing the normality model on the others counts. This will find trains that behave differently from the others with respect to some event type. Alternatively it is possible to find a train that has changed behavior recently, by testing its counts from a recent time period against those from a longer historical time period.

The Anomaly Detection Module

Figure 3 shows the anomaly detection view in Addtrack, which is powered by the anomaly detection module described in this paper. Here, the bars indicate how deviating from normal, i.e. anomalous, each event code count is for each train during the last month. The high bars are much more sparse in this view, and thus easier to go through, compared to those in Figure 1. Also, there are indeed some high bars that corresponds to rather low bars in the count diagram, and which consequentially might have been missed without the anomaly detection. Using the module, analysis can therefore be made much more efficiently by allowing the analyst to focus on a particular subset of trains, event codes and time periods.

Savings from using the anomaly detection module in Addtrack comes mainly from being able to faster diagnose and correct faults on train units in all phases during their lifecycle, which has a number of benefits including a higher ratio of trains delivered on time. The annual savings from using Addtrack are in the order of ten of millions Euro. Out of these, the monetary savings from using the anomaly detection part are substantial but harder to quantify. Bombardier has however estimated that in the long run, maintenance costs can be reduced with between 5% and 10% by using Addtrack with anomaly detection.

In order to simplify the application of anomaly detection for different kinds of log and alarm data in different domains, the anomaly detection algorithm for event data is placed in a separate program module with a very simple API designed to be independent of the specific domain. Since Addtrack runs on the Windows platform, the module was wrapped in a DLL compiled from the C source code, which was then integrated in Addtrack. The API

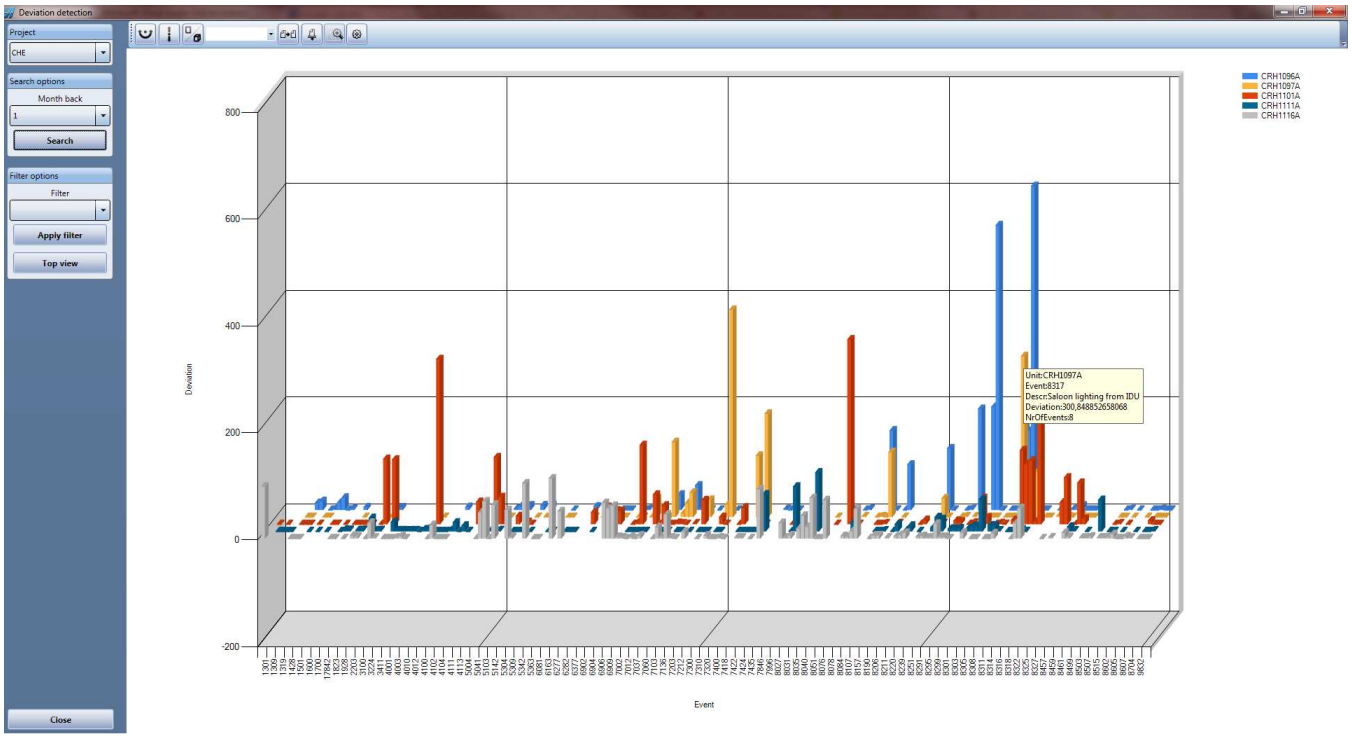

Figure 3: Addtrack deviation detection mode for selected event occurrences for five units in the Chinese fleet (same as in Figure 1).

has only four functions: *CreateAnomalyDetector* which creates a new anomaly detector and trains it with a set of provided event counts and intervals; *ApplyAnomalyDetector* which tests it on some also provided event counts and intervals; *SetAnomalyThreshold* to set a threshold used when filtering out anomalous samples from the training data; and *DeleteAnomalyDetector* to call when the detector is not needed anymore.

This simple design with a minimum of free parameters makes it easy to provide anomaly detection capacity to other programs. To use the anomaly detector, the host program will have to take care of any user interaction to e.g. select a time period of interest and a set of entities to check for anomalies (like trains in the case of this paper); look up from appropriate databases a list of event counts for the different entities during the selected interval; create and apply an anomaly detector to get anomaly values for the entities; and visualize the results in a suitable way to the user. If the surrounding program is a commercial product aimed at analysis of log or event data, it is likely to have functionality for user interaction, data base access, and result visualization already, and it will therefore be relatively simple to add anomaly detection functionality as well.

Development and Deployment Process

Addtrack has been continuously developed for approximately ten years by a small team of developers, first at Bombardier Transportation, and then at Addiva. The need for anomaly detection was first uncovered in a collaborative research project which was initiated in 2003. At this time, Addtrack was used internally at Bombardier under the name of Edgar (the graphical user interface part) and T-Rex (the database part). Early experiments with much simpler anomaly detectors resulted in a high number of false alarms, which is why a Bayesian approach was chosen. The anomaly detection techniques had also been previously developed at SICS since 2001 during several basic and applied research projects. Isolation of the anomaly detection functionality into its own module started in 2007, and have since then taken only a few person months of work, spread out over time. The addition of the anomaly detection module to Addtrack was, in comparison to the development effort, quite small, and took in the order of 3–4 person weeks of interface design for SICS and approximately the same amount of integration work for Addiva. The roll-out of the module was done via an automatic update of Addtrack over the Internet, and distribution and deployment were therefore relatively easy.

The anomaly detection module was officially released by Addiva at the Addtrack user group conference in 2009, which was used as a marketing channel. From there, the users have spread the information further within their respective company. Manuals and other supporting material have been developed by Addiva, and maintenance and further development of the Bayesian module have been done as part of two consecutive collaborative research projects. In the end, Addiva has the responsibility to integrate new versions of the Bayesian module into Addtrack.

Evaluation

To illustrate how the Bayesian anomaly detection approach performs we have used event data from Regina train sets, which are operated by SJ (the largest passenger railway operator in Sweden) and manufactured by Bombardier Transportation. A Regina train set may consist of two or three cars: either a type DMA car coupled with a type DMB car, or a type DMA car followed by a type T0 car and a type DMB car. These different car types have somewhat different profiles in what events are generated. Therefore it is often advisable to compare a car to other cars of the same type when looking for anomalies. The data were collected from 57 Regina train sets running in the Mälar region in Sweden in the period between 2003-10-01 and 2004-04-19. The number of different possible event codes is 1013, of which 695 different codes were registered during the period. The number of events in each sample was between 1 and 1849 with a median of 5 and interval lengths varied between 16440 and 40845 kilometers. The number of samples per event type varied between 1 and 21, with a median of 6 samples per type.

For each of the occurring event codes, each Regina car was compared to all other cars of the same type, over the time period as a whole. The cars displaying the largest anomalies are shown in Table 1. It is then possible to take a closer look at the anomalies of interest by comparing the data for a single week at a time of a car and event code in question against all other cars and weeks. This is shown for the two first entries of Table 1 in Figure 4 and 5.

Car	Event type	Anomaly (Λ)
3029	HVAC communication failure	9316.7
9024	Train heating, fuse failure	8900.6
9004	Incorrect speed, axle 4	5605.0
1002	Large speed difference, axle 3	3213.6
9046	Low cooling water level in converter	3185.0
3044	Incorrect speed, axle 3	3046.3
9052	Large speed difference, axle 2	3024.3
9049	Large speed difference, axle 1	2834.8
1054	Carbon strip supervision disconnected	2371.1
1053	Carbon strip supervision disconnected	2304.7

Table 1: The ten most abnormally occurring event codes.

As can be seen from the figures, many of the detected anomalies above lasted for a long time before they were eventually eliminated. At the same time it is clear that they are detectable already from the first week or even first few days. Using this kind of anomaly detection therefore has the potential of giving important information to the service organization, with the possibility to rectify the problem earlier than today, often requiring less extensive repairs, causing less wear on related components, and giving shorter time with reduced functionality.

We have above mentioned the importance of using Bayesian statistics, rather than classical point estimates of parameters, to limit the false alarm rate. For comparison between the approaches, a simple anomaly detector was implemented based on a point estimate $\hat{\lambda}$ of the Poisson rate

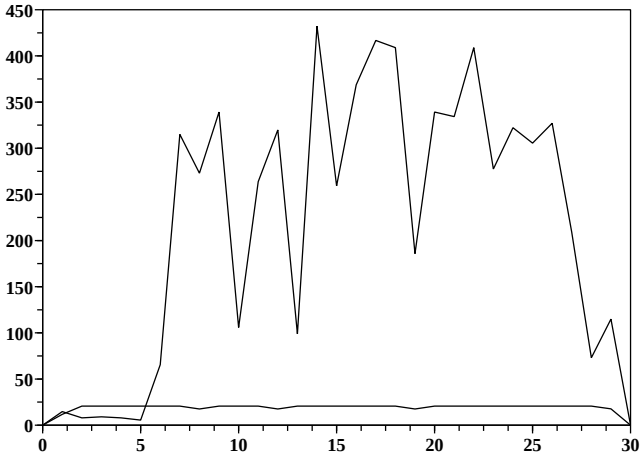

Figure 4: Anomaly per week for the event "HVAC communication failure", for the detected car 3029 (upper curve),

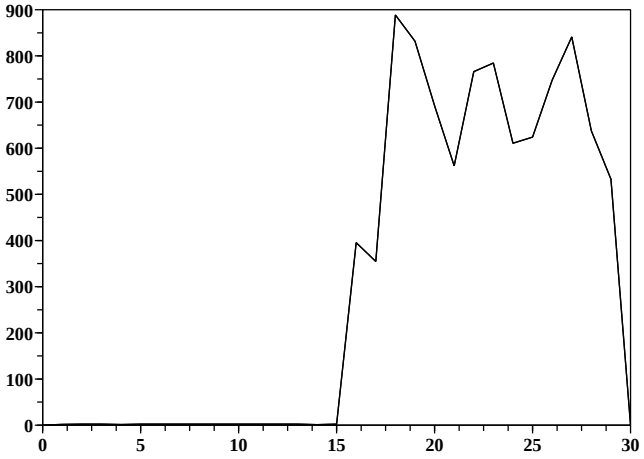

Figure 5: Anomaly per week for the event "Train heating, fuse failure" for car 9024.

per time unit λ together with the principal anomaly defined in Equation (1), using $\theta = \hat{\lambda}T$ for the interval length T. The simple detector uses the maximum likelihood estimate for the Poisson rate, which is equal to the sample mean value for the observed data divided by the interval length:

$$\hat{\lambda} = \frac{1}{n}\sum_{i=1}^{n} x_i/T_i \tag{7}$$

where T_i is the length of the interval for sample i.

An anomaly threshold of $\epsilon = 10^{-6}$ was used in both cases. When comparing the two anomaly detectors, out of 2841 samples, 2483 were classified as normal and 323 as anomalous by both. In addition, 35 samples were classified as anomalous by the point estimate anomaly detector but not by the Bayesian anomaly detector, while no samples were classified as anomalous by the Bayesian anomaly detector but not by the point estimate anomaly detector. In this case,

the Bayesian anomaly detector was therefore more robust with regard to false alarms than the non-Bayesian approach.

An anomaly fraction of 0.11 or 0.13 can be considered high compared to the 10^{-6} which should be expected if the Poisson model assumption holds perfectly, there is no measurement noise, and no anomalies occur. Nonetheless, the Bayesian anomaly detector is quite useful in practice to focus further analysis efforts, which field experience from the use of Addtrack also confirms.

Discussion

There are many challenges when trying to use anomaly detection in a real world application. The Bayesian Principal Anomaly however, has many suitable properties for this: the false alarm rate, which is a major problem for many anomaly detection algorithms when used in practice, can be controlled directly by adjusting the anomaly threshold; the Bayesian approach makes the system work also when there are limited amounts of training data, as is often the case; and the training data used may itself contain anomalies, i.e. it need not be absolutely clean as for some other anomaly detection methods, since the method will itself test each sample and only learn those that are judged non-anomalous.

Anomaly detection of event data is a rather general task that is applicable in a large number of situations. Many systems today generate log messages or alarms, and typically in such volymes that manual inspection is problematic. The risk that an operator misses an important alarm among the large amounts of less important alarms is significant. The chance of manually noticing subtle changes in rates of different alarms is small. The anomaly detection method presented here is not limited to the train domain, but will function on log data from almost any system.

The design of the anomaly detection module, with only a small number of access routines and a minimal amount of free parameters, makes it easy to plug into existing analysis tools for various domains and application areas. This allows the module to go piggyback on existing analysis products, thereby avoiding the overhead of producing, marketing, and supporting a stand-alone anomaly detection product.

References

Bjurling, B.; Holst, A.; Ståhl, O.; and Wallgren, A. 2010. Statistical Anomaly Detection and Visualization. In *Proc. of the 1st National Symposium on Technology and Methodology for Security and Crisis Management*.

Cemerlic, A.; Yang, L.; and Kizza, J. 2008. Network intrusion detection based on bayesian networks. In *Proc. of the 20th International Conference on Software Engineering and Knowledge Engineering*.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly Detection: A Survey. *ACM Computing Surveys* 41(3).

Chebrolu, S.; Abraham, A.; and Thomas, J. 2005. Feature deduction and ensemble design of intrusion detection systems. *Computers & Security* 24(4):295–307.

Dey, C. 2009. Reducing IDS false positives using Incremental Stream Clustering (ISC) Algorithm. Master's thesis, Dept. of Computer and Systems Sciences, Royal Institute of Technology.

García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; and Vázquez, E. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security* 28(1-2):18 – 28.

Hill, D.; Minsker, B.; and Amir, E. 2009. Real-time bayesian anomaly detection in streaming environmental data. *Water Resour. Res* 45:W00D28.

Holst, A., and Ekman, J. 2011. Incremental Stream Clustering for Anomaly Detection and Classification. In Kofod-Petersen A., H. F., and H., L., eds., *Proc. of the 11th Scandinavian Conference on Artificial Intelligence*, 100–107. Trondheim, Norway: IOS Press.

Holst, A.; Ekman, J.; and Larsen, S. 2006. Abnormality detection in event data and condition counters on Regina trains. In *Proc. of the IET International Conference on Railway Condition Monitoring*, 53–56.

Lerner, U.; Parr, R.; Koller, D.; and Biswas, G. 2000. Bayesian fault detection and diagnosis in dynamic systems. In *Proc. of the 12th Innovative Applications of Artificial Intelligence Conference*, 531–537.

Shahabdeen, J. A.; Baxi, A.; and Nachman, L. 2010. Ambulatory energy expenditure estimation: A machine learning approach. In *Proc. of the 22nd Innovative Applications of Artificial Intelligence Conference*.

Su, B., and Xu, C. 2009. Not so naive online bayesian spam filter. In *Proc. of the 21st Innovative Applications of Artificial Intelligence Conference*.